論 文 の 内 容 の 要 旨

論文題目　Automation of Building Malicious Script Analysis Systems for Diverse Execution Environments
(悪性スクリプト解析における多環境対応のための
システム自動構築に関する研究)

氏　名　碓井　利宣

This thesis consists of seven chapters, as summarized above.

In Chapter 1, we introduce the background of this thesis and clarify the problems addressed in this thesis. We first introduce that recent malware (malicious software) used by attackers has not only a form of an executable binary but also diverse forms such as various files with exploit code (exploit files) and scripts with malicious behavior (malicious scripts). This diversity in forms forces malware analysts to build protection systems for each of them respectively, which is almost unrealistic from the perspective of the required human effort. We then show that the existing techniques that analyze and detect these malicious files cannot solve this problem. In the following chapters, we address this problem by introducing analysis and detection techniques applicable to diverse forms of exploit files and diverse languages of malicious scripts.

In Chapter 2, we propose a method for statically detecting ROP chains in malicious data, including malicious files, by learning the target libraries (i.e., the libraries used for ROP gadgets). Our method accelerates inspection by exhaustively collecting feasible ROP gadgets in the target libraries and learning them separated from the inspection step. In addition, we reduce false positives inevitable for existing static inspection by statically verifying whether a suspicious byte sequence can properly link when executed as a ROP

chain. Experimental results on our prototype system called ROPminer showed that our method had achieved millisecond-order ROP chain detection with high precision. Because ROP chains are almost essentially embedded in exploit files, static detection of them, which does not have requirements to deploy, can protect the various endpoints against the diverse form of exploit files.

In Chapters 3-5, the main part of this thesis, we introduce approaches that build malicious script analysis tools of script API tracers, multi-path explorers, and taint analysis frameworks, respectively. The script API tracers log the called script APIs during the execution of the target script. The multi-path explorers execute exhaustive paths in the target script. The taint analysis frameworks enable us to track the data flow in the target script. By automating to build these analysis tools, this thesis provides protection against malicious scripts written in diverse script languages.

In Chapter 3, we propose an approach for detecting the hook and tap points in a script engine binary essential for building a script API tracer. Our approach allows us to reduce the cost of reverse engineering on a script engine binary, which is the largest portion of the development of a script API tracer, and build a script API tracer for a script language with minimum manual intervention. We implemented a prototype system with our approach called STAGER. The experimental results showed that our approach built the script API tracers for the three script languages popular among attackers (VBA, VBScript, and PowerShell). The results also demonstrated that these script API tracers successfully analyzed real-world malicious scripts.

In Chapter 4, we propose an approach that builds multi-path explorers based on vanilla script engines by dynamically analyzing them to obtain architecture information of their VMs required for multi-path exploration. Our approach executes multiple test scripts to obtain execution traces of the target script engine and differentiates them for extracting architecture information of its VM. We implemented a prototype system with our approach called STAGER M and evaluated it with Lua and VBScript. The experimental results showed that our approach could correctly extract the architecture information within a realistic time frame. Using the information, we built multi-path explorers and confirmed that they could effectively analyze real-world evasive malicious

scripts.

In Chapter 5, we propose an approach that builds taint analysis frameworks for scripts on top of the framework designed for native binaries. We first conducted experiments to reveal that the semantic gaps in data types between binaries and scripts disturb our approach by causing under-tainting. To address this problem, our approach detects such gaps and bridges them by generating force propagation rules, which can eliminate the under-tainting. We implemented a prototype system with our approach called STAGER T and evaluated it with Python and VBScript. We built taint analysis frameworks for Python and VBScript with STAGER T and confirmed that they could effectively analyze the data flow of real-world malicious scripts.

In Chapter 6, we provide overall discussion involved in multiple chapters (especially in Chapters 3-5). We first discuss how we can effectively combine the approaches proposed in the chapters and then describe how practical the malicious script analysis systems built by combining them would be. From a different perspective, we also discuss what can be proposed for future script engines with the insight obtained in the chapters. More concretely, we propose to future script engines that they have an interface that provides information helpful for building malicious script analysis systems by extending the design of an existing interface provided by the script engines of Microsoft Corporation.

In Chapter 7, we conclude this thesis by summarizing our contributions and describing the future prospects of the research field newly developed in this thesis.